

A WEB-BASED MIDI CONTROLLER FOR MUSIC LIVE CODING

Frank Heyen Dilara Aygün Michael Sedlmair

VISUS, University of Stuttgart, Germany

frank.heyen@visus.uni-stuttgart.de, michael.sedlmair@visus.uni-stuttgart.de

ABSTRACT

We contribute an interactive visual frontend to live coding environments, which allows live coders and performers to influence the behavior of their code more quickly and efficiently. Users can trigger actions and change parameters via instruments, buttons, and sliders, instead of only inside the code. For instance, toggling a loop or controlling a fading effect through mouse or touch interaction on a screen is faster than editing code. While this kind of control has already been possible with hardware MIDI devices, we provide a more accessible, easy-to-use, and customizable alternative that only requires a web browser. With examples, we show how users perform live-coded music faster and more easily with our design compared to using pure code.

1. INTRODUCTION

Several tools support programming education playfully [1–4], including Sonic Pi [2,3], an environment for music live coding [5] primarily aimed at schools. Instead of implementing mathematical functions and other common exercises, Sonic Pi allows exploring code with the more joyful and motivating task of creating music. Live coding allows to immediately see – or rather hear – the results of changes. However, some musical features are tricky to implement or perform through code, making live coding more lavish than it could be and distracting from more interesting tasks. For example, Sonic Pi’s low-level code interface hinders children in creative composition [4], motivating us to augment it with higher-level controls. We therefore add an interactive visual frontend, allowing to implement and control tricky musical features more easily. For instance, toggling tracks currently requires finding corresponding lines of code and commenting them out, which might take too long and lead to mistakes such as confusing lines. A complex example would be a fading effect: It is hard to quickly perform such an effect or spontaneously change the fading curve. Interactive controls simplify performing above tasks, allowing to focus on creativity and learning rather than implementation. Sonic Pi can be controlled through MIDI [6] hardware such as keyboards and controller pads.

However, these lack accessibility, as many users or schools cannot or do not want to purchase them or might have limited funding and storage – a class of 30 students, each with a MIDI controller would not be sustainable [7]. In contrast, our approach requires no additional hardware, only a computer running Sonic Pi and a web browser, making our interface accessible for all who already have access to Sonic Pi. Combining live coding and visual control could lower the learning curve in programming and music education or for *just* having fun by experimenting with new music interfaces. As we target students and live performers, we keep our design simple and familiar by restricting the complexity of features. Therefore, our Sonic Pi Controller is simpler and more user-friendly than other music interfaces, such as digital audio workstations. In summary, we contribute a simple and user-friendly interactive visual user interface to support users in live coding. We also provide our source code, a web app, and code examples¹.

2. RELATED WORK

Sonic Pi [2,3] is a live coding environment originally designed for programming lessons at schools. Collaborative instruments [8] can be played by multiple people at once or enable musicians with different coding experiences to collaborate and manipulate the others output [9]. Web-based instruments [10,11] allow users to make music, live-code in collaboration [12], or design new instruments [13]. There is research on using web-based instruments for education, such as EarSketch [14] for teaching coding and producing music through a sound browser with recommendations. Another example is a virtual 3D environment [15] with realistic instrument feedback that helps learn scientific and mathematical principles. The web-based gibberwocky [16] live-coding system is similar to our work, but focuses more on features than learnability. Work in the domain of augmented reality explored visually extending hardware MIDI controllers [17], similar to our interface.

3. DESIGN

Our boilerplate code reacts to incoming messages by directly playing notes or samples for instruments, or updating Boolean and floating point variables for buttons and sliders. Live coders can access these variables inside conditionals or effect parameters – or in any other way they like, for example to control custom functions. For imple-

 © F. Heyen, D. Aygün, and M. Sedlmair. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: F. Heyen, D. Aygün, and M. Sedlmair, “A Web-Based MIDI Controller for Music Live Coding”, in *Extended Abstracts for the Late-Breaking Demo Session of the 23rd Int. Society for Music Information Retrieval Conf.*, Bengaluru, India, 2022.

¹ github.com/visvar/sonic-pi-controller

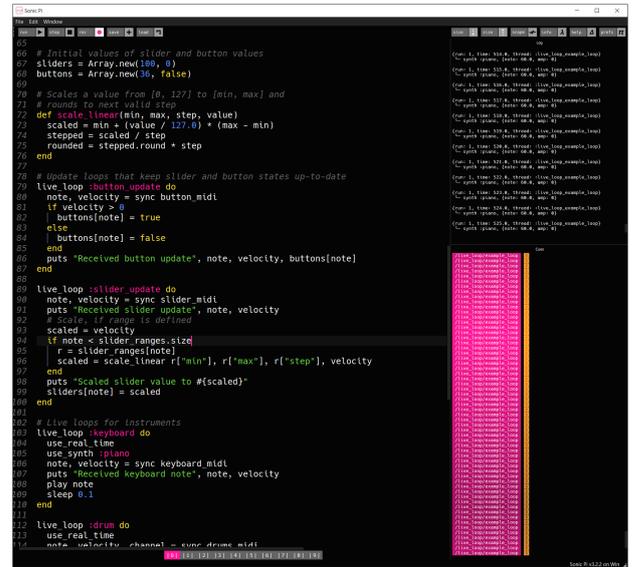
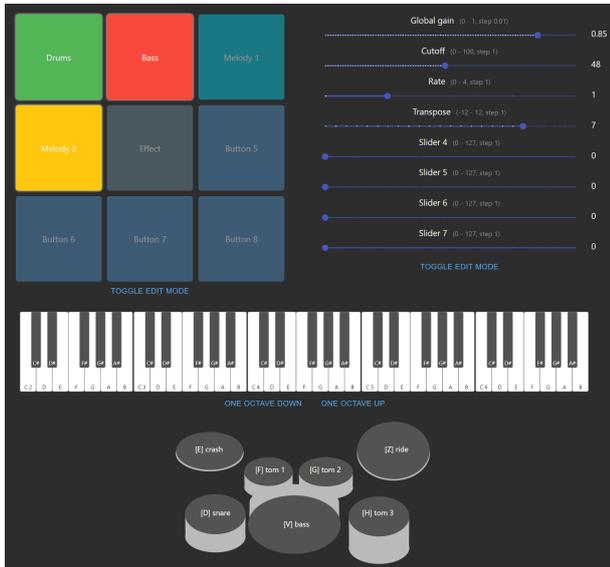


Figure 1. Our design allows live coders to perform certain actions through a visual interface to influence the behavior of code without edits. Commands are transmitted over MIDI from a web app to Sonic Pi.

mentation and live demo, see our supplemental material¹. Users may want to spontaneously *trigger* samples or synthesized melodies while their code still plays in the background, for example when improvising over a drum loop. With code, they have to write down all notes and press play. The temporal disconnect between writing and hearing gets in the way of improvising, resembling composing rather than playing. We implemented two example *instruments*, a piano keyboard and a drum kit, which are played through clicks or taps when using a touch screen. Users can also play the drum with the computer keyboard. With *buttons*, we allow *toggleing* between different sounds and loops, for example a bass loop or low pass filter. In code, this means searching and commenting the loop’s lines, while our frontend allows toggleing by simply pressing a button. We stay close to familiar hardware button matrices [18] and imitate the illumination that shows whether a state is active through opacity. The buttons’ color and label can be customized according to their functionality. In code, each button’s state is synced to a Boolean variable, toggled between true and false by clicking the button. Many effects allow fine-grained *control through parameters*, but changing these quickly and precisely during a performance is hard and limited by typing speed. With live coding, a performer can achieve a fading effect by constantly changing a value over time or implementing fading in code, which is hard². Our *sliders* allow to change any discrete or continuous values, such as volume or effect parameters interactively. For instance, changing a track’s gain or a filter’s threshold requires rapidly updating parameters for a smooth transition. Sliders can be labelled and their ranges adjusted by choosing minimum, maximum, and step values. Interacting with sliders changes integer or float variables which can be accessed in any part of the code. The above tasks are only examples, but artists might find further tasks that could

² See in-thread.sonic-pi.net/t/fading-live-loop-in-out/2464

be supported by custom code or extensions. In our implementation, we chose MIDI for communication over alternatives³, as it is a commonly used, open, and flexible standard. Users can replace or complement our approach by other MIDI controllers at any time, or replace Sonic Pi with other environments, avoiding a lock-in. Moreover, MIDI allows for a web app that users can access without requiring a setup or technical experience. We implemented our frontend as a React app, requiring a browser that implements the Web MIDI API.

4. DISCUSSION

Some limitations affect different features of our approach: Latency and Sonic Pi’s scheduling cause lags between interaction and effect, which is fine for buttons and sliders, but requires users to adapt their timing when improvising. This limitation applies to all hardware or software MIDI inputs. Compared to hardware, our approach lacks haptics: blindly turning knobs is not possible. Instead, our design is free and more adaptable, making it viable for education and novices, while experienced long-term users can switch to or additionally use hardware any time, as there is no lock-in. We show states of buttons and sliders, but no feedback for what happens after commands, which could be interesting future work, such as a piano roll of all played notes that provides an overview of what is happening. A more advanced feature would be a visual representation of envelopes that users adjust by dragging. While creating music through code is great for experimenting and learning, those features would help users *see* their performance.

5. ACKNOWLEDGEMENTS

This work was funded by the Cyber Valley Research Fund.

³ For example [sonic-pi-tool](https://github.com/lpil/sonic-pi-tool) (github.com/lpil/sonic-pi-tool) and [sonic-pi-cli](https://github.com/Widdershin/sonic-pi-cli) (github.com/Widdershin/sonic-pi-cli)

6. REFERENCES

- [1] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The Scratch programming language and environment," *ACM Transactions on Computing Education (TOCE)*, vol. 10, no. 4, pp. 1–15, 2010.
- [2] S. Aaron, "Sonic Pi – performance in education, technology and art," *International Journal of Performance Arts and Digital Media (IJPADM)*, vol. 12, no. 2, pp. 171–178, 2016. [Online]. Available: <https://doi.org/10.1080/14794713.2016.1227593>
- [3] S. Aaron, A. F. Blackwell, and P. Burnard, "The development of Sonic Pi and its use in educational partnerships: Co-creating pedagogies for learning computer programming," *Journal of Music, Technology & Education (JMTE)*, vol. 9, no. 1, pp. 75–94, 2016.
- [4] C. Ford, N. Bryan-Kinns, and C. Nash, "Creativity in children's digital music composition," in *International Conference on New Interfaces for Musical Expression (NIME)*, 4 2021, <https://nime.pubpub.org/pub/ker5w948>. [Online]. Available: <https://nime.pubpub.org/pub/ker5w948>
- [5] N. Collins, A. McLean, J. Rohrerhuber, and A. Ward, "Live coding in laptop performance," *Organised Sound*, vol. 8, no. 3, pp. 321–330, 2003.
- [6] R. A. Moog, "MIDI: Musical instrument digital interface," *Journal of the Audio Engineering Society (AES)*, vol. 34, no. 5, pp. 394–404, 1986.
- [7] R. Masu, A. P. Melbye, J. Sullivan, and A. R. Jensenius, "NIME and the environment: Toward a more sustainable NIME practice," in *International Conference on New Interfaces for Musical Expression (NIME)*, 4 2021, <https://nime.pubpub.org/pub/4bb15lod>. [Online]. Available: <https://nime.pubpub.org/pub/4bb15lod>
- [8] S. W. Lee and G. Essl, "Live coding the mobile music instrument." in *International Conference on New Interfaces for Musical Expression (NIME)*, 2013, pp. 493–498.
- [9] A. Sarwate, R. Rose, J. ARMITAGE, J. Freeman *et al.*, "Performance systems for live coders and non-coders," 2018. [Online]. Available: http://www.nime.org/proceedings/2018/nime2018_paper0082.pdf
- [10] C. Roberts, G. Wakefield, and M. Wright, "The web browser as synthesizer and interface," in *International Conference on New Interfaces for Musical Expression (NIME)*, 2013, pp. 313–318. [Online]. Available: https://www.nime.org/2013/program/papers/day2/paper6/282/282_Paper.pdf
- [11] L. Wyse and S. Subramanian, "The viability of the web browser as a computer music platform," *Computer Music Journal (CMJ)*, vol. 37, no. 4, pp. 10–23, 2013.
- [12] C. McKinney, "Quick live coding collaboration in the web browser," in *International Conference on New Interfaces for Musical Expression (NIME)*, B. Caramiaux, K. Tahiroglu, R. Fiebrink, and A. Tanaka, Eds. nime.org, 2014, pp. 379–382. [Online]. Available: http://www.nime.org/proceedings/2014/nime2014_519.pdf
- [13] C. Roberts, G. Wakefield, M. Wright, and J. Kuchera-Morin, "Designing musical instruments for the browser," *Computer Music Journal (CMJ)*, vol. 39, no. 1, pp. 27–40, 2015.
- [14] J. Smith, M. Jacob, J. Freeman, B. Magerko, and T. Mcklin, "Combining collaborative and content filtering in a recommendation system for a web-based DAW," in *Proc. of the International Web Audio Conference (WAC)*, 2019.
- [15] K. Kritsis, A. Gkiokas, C. A. Acosta *et al.*, "A web-based 3D environment for gestural interaction with virtual music instruments as a STEAM education tool," in *International Conference on New Interfaces for Musical Expression (NIME)*, 2018, pp. 348–349. [Online]. Available: https://www.nime.org/proceedings/2018/nime2018_paper0075.pdf
- [16] C. Roberts and G. Wakefield, "Gibberwocky: new live-coding instruments for musical performance." in *International Conference on New Interfaces for Musical Expression (NIME)*, 2017, pp. 121–126.
- [17] A. Jones and F. Berthaut, "ControllAR: Appropriation of visual feedback on control surfaces," in *Proceedings of the ACM International Conference on Interactive Surfaces and Spaces (ISS)*. ACM, 2016, pp. 465–468.
- [18] B. Rossmly and A. Wiethoff, "Musical grid interfaces: Past, present, and future directions," in *International Conference on New Interfaces for Musical Expression (NIME)*, 4 2021, <https://nime.pubpub.org/pub/grid-past-present-future>. [Online]. Available: <https://nime.pubpub.org/pub/grid-past-present-future>